A Personal Assistant for Health Care Professionals based on Clinical Protocols

Tiago Oliveira[1], António Silva[2], José Neves[1], and Paulo Novais[1]

Algoritmi Research Centre/Department of Informatics, University of Minho
Braga, Portugal
[1]{toliveira,pjon,jneves}@di.uminho.pt
[2]pg25307@alunos.uminho.pt

Abstract. Current tools to operationalize Computer-Interpretable Guidelines focus mainly on displaying recommendations rather than assisting health care professionals in their daily activities. Furthermore, their underlying models have limitations at the level of temporal representation that hinder the accurate depiction of clinical protocols in a few specific situations. This work identifies such situations and proposes a comprehensive temporal model based on Ontology Web Language (OWL), along with a web-based tool that provides an alternative way to deploy and view clinical protocols. This is evaluated through a case study featuring a clinical protocol for the treatment of colon cancer. It was possible to observe that the model was able to represent the majority of temporal patterns, specially those with periodic events and temporal restrictions about the state of a patient.

1 Introduction

Keeping track of their patients is a laborious task for health care professionals, not only because of the number of patients they tend to, but also due to the complexity of the procedures they have to apply. Clinical Decision Support Systems (CDSSs) that provide patient-specific recommendations may help to ease the burden on health care professionals, but they lack functionalities that would allow them to become more prominent in daily clinical practice, namely those that enable: patient tracking, patient follow-up, scheduling of procedures, and monitoring of procedure constraints [5]. Systems that implement them are available, yet there is an absence of integrated solutions that combine these functionalities with traditional CDSS tasks such as diagnosis and treatment recommendation.

The present work discloses one such solution, the CompGuide web application, based on digital versions of clinical protocols for automatic interpretation, also known as Computer-Interpretable Guidelines (CIGs) [4,7]. The underlying model for CIGs used in this work explores Ontology Web Language (OWL) as the support for the definition of representation primitives and the procedural logic of clinical protocols. The application performs the role of a personal assistant for health care professionals that provides decision support and treatment

recommendations, reminders for the timely execution of clinical tasks, and notifications about starting time, ending time, and expected outcomes of tasks. To do so, the temporal representation of clinical tasks is the main aspect to take into account and the main subject of this work.

The present article is organized as follows. Section 2 provides related work about the temporal representation of tasks in CIG models. The underlying CIG model and the temporal representation are disclosed in section 3. Section 4 describes a case-study used to assess the expressiveness of the model and the approach followed to make protocols represented according to it available for execution. Finally, section 5 presents conclusions about the work developed so far and future work considerations.

2 Temporal Representation of Clinical Protocols

From the analysis of the main CIG models [9,10,3,1,8], it was possible to divide temporal constraints of clinical protocols into two groups: temporal constraints placed on the execution of clinical tasks and temporal constraints on conditions about the state of a patient. The analysed models, except for Arden Syntax [8], follow a Task Network Model (TNM) in which every clinical recommendation is considered a task. The first group includes temporal patterns that determine how tasks should be executed, namely: durations, which express how long a task should last; repetitions, the number of times a clinical task should be performed over time; periodicities, which express that a task should be performed from time to time, as a succession of several events; waiting times, delays in the execution of tasks; and repetition conditions, conditions about the state of a patient that determine whether a task should be repeated or not. The second group consists of temporal constraints that reflect changes occurred, or expected to occur, in the state of a patient. It is possible to observe in Table 1 that each model shows at least one limitation in one type of temporal constraint. While the duration and waiting time patterns are present in most models, it is only possible to define an important pattern such as periodicities in three of them, and the same goes for repetition conditions. That being said, the GLARE [1] model is specialized in the representation of periodic procedures and is the most comprehensive of the lot. Another drawback of current CIG models is they do not provide adequate representation primitives for temporal constraints regarding conditions about the state of a patient.

In order to become operational, CIGs need an execution engine to interpret the protocol and a tool through which recommendations are conveyed to health care professionals and information is fed to the engine. Tools such as the Guideline Execution Engine (GLEE), SAGEDesktop, or the execution engine of GLARE [4], to name a few, are used to interact with medical personnel. However, they usually do so by displaying protocols as oriented graphs, with no intelligent integration of the recommendations provided by the protocol in the daily schedule of health care professionals.

Table 1. Assessment of CIG models. The symbol ✓indicates the model fully represents the temporal constraint and the ✗ indicates the model does not represent it or has limitations regarding it.

CIG Model	**Temporal restrictions about the execution of tasks**					**Temporal restrictions about the state of a patient**
	Durations	Repetitions	Periodicities	Waiting Times	Repetition Conditions	
Arden Syntax [8]	✓	✗	✗	✓	✗	✗
GLIF3 [3]	✓	✗	✗	✗	✗	✓
Asbru [9]	✓	✓	✓	✓	✗	✗
PROforma [10]	✓	✓	✗	✓	✓	✗
GLARE [1]	✓	✓	✓	✓	✓	✗

3 Proposed Temporal Model

Following the limitations identified in section 2, one proposes a comprehensive temporal model. The definition of temporal representation primitives follows the guiding principles of the CompGuide ontology for clinical protocols [6]. This model is based on OWL and provides representation primitives for *Plans*, *Actions*, *Questions*, and *Decisions*. Typically, *Actions* describe tasks that should be carried out by the health care professional, such as exams, observations and so forth. *Questions* are used to feed information about clinical parameters to the execution engine and derive new task recommendations. *Decisions* use that information to perform diagnosis or update the state of the patient. Finally, *Plans* contain instances of any other type of task and are defined to achieve specific goals. CompGuide also provides representation primitives to define different types of conditions, including trigger conditions to select one amongst alternative tasks, pre-conditions to execute tasks, and expected outcomes from tasks. The *Condition* class allows the representation of these conditions with specific properties for clinical parameters and their values. The classes of the temporal model are shown in Figure 1. The main classes are represented as subclasses of *TemporalElement*. One of those subclasses is *TemporalUnit* which represents the different units in which a temporal constraint may be expressed. It is an enumerated class including the instances *second*, *minute*, *hour*, *day*, *week*, *month*, and *year*.

3.1 Temporal Constraints on the Execution of Tasks

The tasks for which it is possible to express durations are *Actions* and *Plans*, since they are the only ones that may unfold over time. The attributes characterizing the *Duration* class are encoded as necessary conditions in OWL (as it is the case with all the other classes). As such, to define a duration, one should choose either to define a maximal and minimal duration, trough the *maxDurationValue* and *minDurationValue* data properties, or to define an exact value for

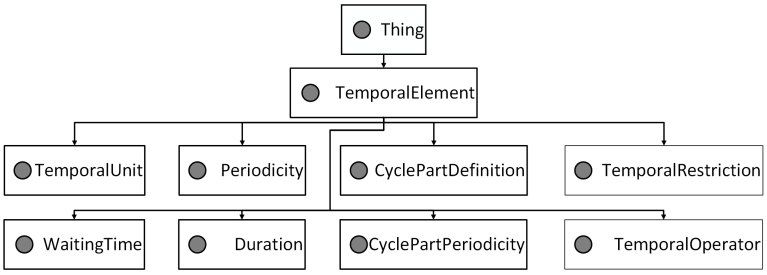

Fig. 1. Classes of the model for the representation of temporal constraints in the CompGuide ontology.

the duration, through the *exactDurationValue* data property. The range of these data properties is defined as a decimal numerical value. Regardless of the type of value one defines, it is always necessary to define a temporal granularity for a decision, which is done through the *hasTemporalUnit* object property connecting instances of *Duration* to instances of *TemporalUnit*.

One can express delays between tasks, motivated for instance by the need to observe the effect a task has on the state of a patient, with the *WaitingTime* class. The waiting time values are defined much like in *Duration*, as intervals (with the *maxWaitingTime* and *minWaitingTime* data properties) or exact values (with the *exactWaitingTime* data property). The *hasTemporalUnit* property is used again to specify the units.

The representation of periodic tasks is the most complex pattern. They are represented with the class *Periodicity*. A periodicity can be defined for any type of task, *Plans*, *Actions*, *Questions* or *Decisions*. However, the periodic event is bound by either a duration, a repetition constraint or a stop condition about the state of the patient. The duration is defined through the reuse of the *Duration* class. As such, an instance of *Periodicity* can also be connected to an instance of *Duration* through the *hasDuration* object property, thus determining for how long a periodic task should take place. On the other hand, if one wants to state the number of times the event should be carried out (the same is to say the number of cycles of the periodic task), it is necessary to formulate a repetition constraint, which is possible through the *repetitionValue* data property, with a range of integer numerical values. Alternatively, it could be the case the periodic task should only occur until a condition about the state of a patient is met. To express this, one uses the *hasStopCondition* object property to connect an instance of periodicity to instances of the class *Condition*. While it is possible for a periodicity to have a duration and a stop condition, a repetition value and a stop condition, or just a stop condition, it is not possible to have both a duration and a repetition value because it is considered to be redundant information. With a duration and a frequency it is already possible to calculate the number of repetitions of a task and vice versa. The stop condition takes precedence over the other temporal restrictions, so, if the condition is met, the task is immediately stopped.The frequency of the periodicity and the temporal granularity are

defined in the data property *periodicityValue* and through the *hasTemporalUnit* object property respectively.

A periodic task unfolds in a series of executions which are handled by the execution engine as events. Each event may have itself an associated periodicity or duration, which means that it may be necessary to define nested temporal patterns. The object property *hasCyclePartDefinition* is used to specify the duration or the periodicity of an event. It connects instances of *Periodicity* to instances of *CyclePartDefinition*. Instances of this class may have the *hasDuration* property connecting them to instances of *Duration* or the *hasCyclePartPeriodicity* property connecting them to instances of *CyclePartPeriodicity*. The latter is similar to *Periodicity* in all but the possibility to define another periodicity or duration within it. One can argue it would be simpler to reuse the *Periodicity* class rather than defining another class for the periodicity of each event, but by doing so it would be possible to nest periodicities inside one another infinitely, which would be difficult to handle computationally.

3.2 Temporal Constraints on the State of a Patient

In CompGuide, a temporal constraint for conditions about the state of a patient is represented by an instance of the *TemporalRestriction* class. To connect the constraint to an instance representing a condition, it is necessary to use the *hasTemporalRestriction* property, which, although non-mandatory, can be defined for any of the above-mentioned conditions.

For each instance of *TemporalRestriction* it is necessary to specify a temporal operator through the *hasTemporalOperator* object property. This object property points to instances belonging to *TemporalOperator*. This is an enumerated class that can only have a limited number of instances, namely *within_the_last* and *within_the_following*. The temporal operators represent the reach of a temporal constraint and are used together with temporal units, defined through the *hasTemporalUnit* object property, and temporal restriction values. The latter are expressed through data properties such as *maxTemporalRestrictionValue* and *minTemporalRestrictionValue* for an interval, or *temporalRestrictionValue* for an exact value, with a range of decimal numerical values.

Each operator conveys a different meaning.The operator *within_the_last* is used when one wants to express that a condition must have held true at least once, within a period of time just before execution time. The execution engine interprets this operator by checking if, in the state of the patient, there is a record regarding the parameter in the condition, registered within the specified time frame, and if its value validates the condition. This temporal operator can be defined for temporal restrictions of simple conditions in *Decision* tasks, trigger conditions and pre-conditions, ant it is used to reason about past events. However, in an expected outcome of a task, it is necessary to express a condition about the future, in which one aims to observe the effect a clinical task has after being applied to a patient. The *within_the_following* operator conveys this meaning to the execution engine which, in turn, checks if the condition holds true after the specified time.

4 Discussion and Implementation

The CompGuide temporal model was validated with a case-study featuring a National Comprehensive Cancer Network (NCCN) protocol for the treatment of colon cancer [2]. This protocol includes procedures that unfold over different phases of treatment, from cancer staging to follow-up, and presents a wide variety of temporal patterns. The representation of the clinical protocol in the model was carried out using Protégé[1], an ontology editor for OWL.

4.1 Analysis of a Case-study in Colon Cancer Treatment

The representation of the NCCN protocol resulted in an *owl* file containing 223 task instances, of which: 190 were *Action* tasks, 21 were *Question* tasks, 1 was a *Decision* task and 11 were *Plans*. Out of the 223 tasks, a total of 95 had temporal constraints. The set included: 7 with *Durations*, 2 with *WaitingTimes*, 79 with *Periodicities*, and 7 with nested *Periodicities*. *Periodicities* were the most abundant pattern, mainly because of the rich description of chemotherapy regimens made in the document. Most *Periodicity* instances were limited by a *Duration*.

The proposed temporal constructors were effective in the representation of the different temporal patterns, specially in the tasks having a *Duration* or a *Waiting Time*. In fact, the information about the duration of tasks was mostly conveyed using exact values or intervals, like what happens in the natural language expression "perform neoadjuvant therapy for 2-3 months", extracted from the protocol [2], in which there is an *Action* consisting in therapy before treatment with a *Duration* expressed using the *minDurationValue* 2.0, the *maxDurationValue* 3.0, and the *TemporalUnit month*. The same is true for waiting times, as seen in the example "reevaluation for colon surgery 2 months after the end of chemotherapy" [2] in which there are clearly two *Actions*, the first is chemotherapy and the second is re-evaluation, with the latter having a delay expressed with the *exactWaitingTime* 2.0, and the *TemporalUnit month*. This temporal model follows a simpler scheme than Asbru which provides a plethora of temporal annotations such as earliest and latest, starting and ending, shifts for tasks [9]. Yet, the CompGuide temporal elements were sufficient to represent all the durations and waiting times of such a complex protocol.

Regarding periodic tasks, as mentioned above, most of them were bounded by a *Duration*. The constraints followed a structure similar to the one in the recommendation "complete physical exam every 6 months for 2 years" [2]. In the example, it is possible to identify the *Action* complete physical exam, the *periodicityValue* 6.0, the *TemporalUnit* for the *Periodicity month*, the *exactDurationValue* 2.0, and the *TemporalUnit* for the *Duration year*. In this case, the execution engine would recommend the execution of the task with the specified frequency during the 2 years. Periodic tasks bounded by the number of repetitions were not that common in the protocol, but their interpretation logic follows

[1] Available at `http://protege.stanford.edu/`.

the same principles as periodic tasks with durations, the execution engine would count the number of times the periodic event was executed and would recommend the task the number of times still left to complete execution. The periodic tasks that had stop conditions usually had a duration limiting their execution. The pattern followed a structure identical to the example "perform colonoscopy every 3 months for 2 years and stop if signs of adenoma are found" [2]. The *Periodicity* identified in the example has the *periodicityValue* 3.0, the *TemporalUnit month*, a *Duration* with the *exactDurationValue* 2.0, and the *TemporalUnit year*. It also has the *Condition* signs of adenoma. After each event of the periodic task, the execution engine should ask the user if the stop condition holds, and, if so, the task is stopped and the execution engine moves on to the following tasks.

The only examples of nested periodicities referred to the description of how the different chemotherapy schemes should be applied. For instance, the expression "CapeOx should be applied every 3 months, with the administration of capecitabine every 12 hours for 14 days" describes an *Action* consisting in applying the CapeOx chemo, that has a *Periodicity* with the *periodicityValue* 3.0, and the *TemporalUnit month*, with a *CyclePartDefinition* which, in turn, has a *CyclePartPeriodicity*. This instance of *CyclePartPeriodicity* has the *periodicityValue* 12.0, and the *TemporalUnit hour*, along with the *Duration* 14 days. This type of constraint tells the execution engine that the event of the task occurring every 3 months should, itself, be performed every 12 hours during 14 days. Once the event is over, it should only be performed again after 3 months.

In the protocol there were 6 occurrences of temporal constraints on conditions about the state of the patient. Most of them expressed the expected outcomes of chemotherapy, as in the expression "the tumor should become operable after 6 months of FOLFOX or CapeOx chemotherapy". Here, an outcome is expressed in the form of the *Condition* the tumor becomes operable, and a *TemporalRestriction* is defined for that condition with the *TemporalOperator within_the_following*, the *temporalRestrictionValue* 6.0, and the *TemporalUnit month*. The execution engine interprets the restriction by checking, after the specified time, whether the outcome was validated and notifies the user of the result. Another common situation was the verification of incompatibilities of chemotherapy regimens. An example of such a situation is the recommendation "for therapy after third progression consider experimental chemotherapy, if the regorafenib regimen has been applied within the last 12 months" which describes the *Action* apply experimental chemotherapy, associated to a trigger condition that determines its selection. The *Condition* is regorafenib having been applied, and there is a *TemporalRestriction* that goes with it, defined with the *TemporalOperator within_the_last*, the *temporalRestrictionValue* 12.0, and the *TemporalUnit month*.

With the examples provided above, it is possible to conclude that the CompGuide temporal model is more encompassing than the existing approaches. When it comes to durations, waiting times, and periodicities, it performs at the level of GLARE [1]. When comparing with the approaches mentioned in Table 1, the examples having periodicities would not have been represented in at least three

of the models. Additionally, the CompGuide model provides a set of primitives for the representation of constraints on conditions about the state of the patient, which are absent from the current approaches, except for one. However, there are some limitations, namely the level of nesting of periodicities and the expression of alternative temporal constraints, related with the freedom of speech allowed in guidelines. For instance, a recommendation may specify that an action should be executed a certain number of times or during a certain period, but, in the model, it is not possible to represent this alternative.

4.2 Protocol Execution and Visualization

The system set up to execute clinical protocols is depicted in Fig. 2. It consists of a *Core Server* that has four distinct components: the *Authentication* component, responsible for authenticating the user into the system; the *Database Handler* to manage the access to the *Database* containing information about physician and patient profiles, patient states, and protocol executions; the *GuidelineHandler* which manages the access to recommendations of clinical protocols in a *Guideline Repository* consisting of a collection of *owl* files; and the *Guideline Execution Engine* which interprets the clinical protocol, analyses the state of the patient, and provides recommendations in the form of tasks. The constraints, including temporal constraints, are defined directly in the ontology, and Semantic Web Rule Language (SWRL) is not used for this specification due to the flexibility and complexity required for temporal constraints.These features are made available by the *Core Server* as RESTful web services in order to ensure they can be easily integrated into any type of application. The Core Server is implemented in Java, using the RESTEasy API over a WildFly Application Server.

The personal assistant, which uses the web services available in the *Core Server*, was developed as a web application following the Model-View-Control (MVC) paradigm using Java Server Faces (JSF). The main interfaces are shown in Fig. 3. The personal assistant provides tasks based on the automated validation of conditions regarding the state of the patient and builds a schedule for the health care professional. As seen in Fig. 3 a), it provides a calendar view of the clinical tasks with different granularities (day, week and month), which can be transformed into a temporal axis view, as seen in Fig. 3 b). The former is intended to provide an overall picture of the tasks that lie ahead, while the latter allows the health care professional to focus on a smaller set of tasks at a time. The application also provides notifications about the different temporal constraints of tasks, alerting the user to when he should execute them, when they should start, when they are due, and the results of expected outcomes. These notifications are shown as side messages, as displayed in Fig. 3 a), and are gathered in a notification stack. By clicking on a task, a panel is shown with task details, namely its description, remaining time and remaining repetitions.

5 Conclusions and Future Work

By building a system that revolves around the temporal model and integrates recommendations into the daily practice of health care professionals, it is pos-

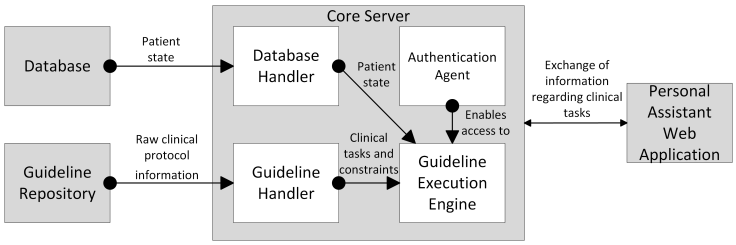

Fig. 2. Architecture of the system for the temporal execution of clinical protocols.

sible to build a schedule for them to follow, endowed with notification features about the correct time to enact clinical tasks. The system displayed herein is innovative in the sense that it is unlike other CIG execution tools. It maps clinical protocols, as they are being applied, onto an agenda. The impact of this is that it enables medical personnel to keep a better track of the clinical processes they are managing. This work aims to disclose the innovative view CompGuide brings to the execution of CIGs its technical feasibility. Additional experiments are in development to evaluate the expressiveness of the model with a wide variety of clinical protocols and a thorough comparison with the other models. In terms of additional features, by accessing other calendar services, it would be possible to fit these tasks into the other activities of the health care professinal's life and possibly sort out schedule conflicts.

Acknowledgements

This work has been supported by FCT Fundação para a Ciência e Tecnologia within the Project Scope UID/CEC/00319/2013. The work of Tiago Oliveira is supported by a FCT grant with the reference SFRH/BD/85291/ 2012.

References

1. Anselma, L., Terenziani, P., Montani, S., Bottrighi, A.: Towards a Comprehensive Treatment of Repetitions, Periodicity and Temporal Constraints in Clinical Guidelines. Artificial Intelligence in Medicine 38(2), 171–195 (2006)
2. Benson, A., Bekaii-Saab, T., Chan, E., Chen, Y.J., Choti, M., Cooper, H., Engstrom, P.: NCCN Clinical Practice Guideline in Oncology Colon Cancer. Tech. rep., National Comprehensive Cancer Network (2013), `http://www.nccn.org/professionals/physician_gls/f_guidelines.asp`
3. Boxwala, A.a., Peleg, M., Tu, S., Ogunyemi, O., Zeng, Q.T., Wang, D., Patel, V.L., Greenes, R.a., Shortliffe, E.H.: GLIF3: A Representation Format for Sharable Computer-Interpretable Clinical Practice Guidelines. Journal of Biomedical Informatics 37(3), 147–61 (Jun 2004)
4. Isern, D., Moreno, A.: Computer-based Execution of Clinical Guidelines: a Review. International Journal of Medical Informatics 77(12), 787–808 (2008)
5. Musen, M.A., Shahar, Y., Shortliffe, E.H.: Clinical decision-support systems. In: Shortliffe, E., Cimino, J. (eds.) Biomedical Informatics, pp. 698–736. Health Informatics, Springer New York (2006)

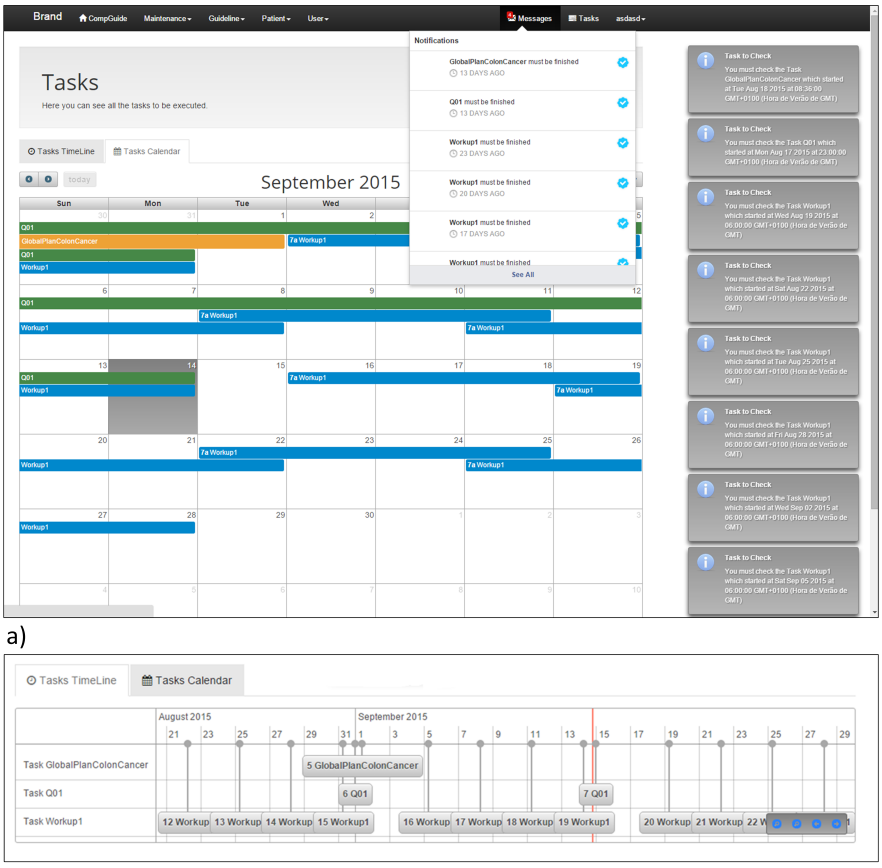

Fig. 3. Clinical task visualization: a) calendar view, notification messages and notification stack; b) temporal axis view.

6. Oliveira, T., Novais, P., Neves, J.: Representation of Clinical Practice Guideline Components in OWL. In: Trends in Practical Applications of Agents and Multiagent Systems SE - 10, Advances in Intelligent Systems and Computing, vol. 221, pp. 77–85. Springer International Publishing (2013)
7. Peleg, M.: Computer-interpretable Clinical Guidelines: A Methodological Review. Journal of Biomedical Informatics 46(4), 744–63 (2013)
8. Samwald, M., Fehre, K., de Bruin, J., Adlassnig, K.P.: The Arden Syntax standard for clinical decision support: Experiences and directions. Journal of biomedical informatics (2012)
9. Shahar, Y., Miksch, S., Johnson, P.: The Asgaard Project: A Task-specific Framework for the Application and Critiquing of Time-oriented Clinical Guidelines. Artificial intelligence in Medicine 14(1-2), 29–51 (1998)
10. Vollebregt, A., ten Teije, A., van Harmelen, F., van der Lei, J., Mosseveld, M.: A study of PROforma, a development methodology for clinical procedures. Artificial Intelligence in Medicine 17(2), 195–221 (1999)